AD-A099 103 STANFORD UNIV CA DEPT OF COMPUTER SCIENCE F/G 12/1
HUFFMAN'S ALGORITHM VIA ALGEBRA,(U)
MAR 81 D E KNUTH N00014-76-C-0330
UNCLASSIFIED STAN-CS-81-841 NL

March 1981

Report No. STAN-CS-81-841

LEVEL II

# Huffman's Algorithm via Algebra

by

Donald E. Knuth

Contract N00014-76-C-0330
NSF-MCS77-23738

Research sponsored by

National Science Foundation
and
Office of Naval Research

Department of Computer Science

Stanford University
Stanford, CA 94305

# Huffman's Algorithm via Algebra

Donald E. Knuth
Computer Science Department
Stanford University
Stanford, California 94305

**Abstract.** The well known algorithm of David A. Huffman for finding minimum redundancy codes has found many diverse applications, and in recent years it has been extended in a variety of ways. The purpose of this note is to discuss a simple algebraic approach that seems to fit essentially all of the applications of Huffman's method that are presently known.



This research was supported in part by National Science Foundation grant MCS-77-23738 and by Office of Naval Research contract N00014-76 C-0330. Reproduction in whole or in part is permitted for any purpose of the United States government.

## Huffman's Algorithm via Algebra

The well known algorithm of David A. Huffman [5] for finding minimum redundancy codes has found many diverse applications, and in recent years it has been extended in a variety of ways [1,2,3,4,6,7,8]. The purpose of this note is to discuss a simple algebraic approach that seems to fit essentially all of the applications of Huffman's method that are presently known. The ideas to be presented are slight extensions and corrections of the results in [8].

We consider a linearly ordered set $A$ on which a binary operator has been defined satisfying the following five axioms:

**A0** (Increasing property). $a \le a \circ b$.

**A1** (Commutative law). $a \circ b = b \circ a$.

**A2** (Cousin law). $(a \circ b) \circ (c \circ d) = (a \circ c) \circ (b \circ d)$.

**A3** (Preservation of order). If $a < b$ then $a \circ c \le b \circ c$.

**A4** (Associative inequality). If $a \le c$ then $(a \circ b) \circ c < a \circ (b \circ c)$.

Given element $a_1, \ldots, a_n \in A$, not necessarily distinct, an *expression* on $\{a_1, \ldots, a_n\}$ is a formula that computes another element of $A$ be applying the binary operation $n-1$ times and using each $a_i$ exactly once. For example, the commutative law A1 states that both of the possible expressions on $\{a, b\}$ have the same value, and the cousin law A2 states that two particular expressions on $\{a, b, c, d\}$ are equal. Expressions are essentially binary trees having $\{a_1, \ldots, a_n\}$ as "leaves".

Huffman's algorithm forms an expression on $\{a_1, \ldots, a_n\}$ in the following way: If $n > 1$, let $a_i$ and $a_j$ be the smallest and second-smallest elements; replace $a_i$ and $a_j$ by $(a_i \circ a_j)$ and repeat the construction on the remaining $n-1$ elements, until eventually $n = 1$.

For example, suppose $A$ consists of the nonnegative integers, and let $a \circ b = 2(a + b)$. It is easy to check that axioms A0–A4 hold. Huffman's algorithm applied to the elements $\{1, 3, 5, 7, 9\}$ will produce the expression

$$\big((5 \circ 7) \circ \big((1 \circ 3) \circ 9\big)\big) = 116.$$

Note that we have

$$5 \cdot 2^2 + 7 \cdot 2^2 + 1 \cdot 2^3 + 3 \cdot 2^3 + 9 \cdot 2^2 = 116;$$

in the case of this particular operation the value is $\sum a_i \cdot 2^{l_i}$, where $l_i$ is the "level" at which $a_i$ appears in the formula, i.e., the depth of parenthesis nesting when parentheses surround each use of the binary operation.

When a binary operator satisfies the commutative and cousin laws A1 and A2, the value of any expression on $\{a_1, \ldots, a_n\}$ depends only on the $a_i$ and their levels $l_i$; in other words, any two expressions in which each $a_i$ appears on a given level $l_i$ will be equal. We can prove this by using terminology from family trees: If '$(a \circ b)$' appears in some formula we can say that $a$ and $b$ are brothers and $(a \circ b)$ is their father. Two elements $a_i$ and $a_j$ on the same level in some expression are either brothers, or they are cousins (their fathers are brothers), or they are second cousins (their fathers are cousins), etc. We can transform the expression to an equivalent one using A1 and A2 until $a_i$ and $a_j$ are brothers; for if $a_i$ and $a_j$ are $k$-th cousins and $k > 1$, the axioms change cousins with brothers, while if $k = 1$ the transformation for order $k-1$ will make their fathers into brothers and one more step will complete the job. Now let $E$ and $E'$ be expressions in $\{a_1, \ldots, a_n\}$ for which the levels $l_i = l'_i$ agree for all $i$. If $n = 1$, clearly $E = E'$. Otherwise $E$ contains some operation $(a_i \circ a_j)$. Since $l_i = l_j$, we can transform $E'$ to an expression $E'' = E'$ in which $a_i$ and $a_j$

This paper is dedicated to Marshall Hall, Jr., on the occasion of his retirement from teaching.

are brothers. Replacing $(a_i \circ a_j)$ by a new symbol in $E$ and $E''$ yields an expression in $n-1$ elements having corresponding level numbers equal, hence $E = E''$.

The main feature of Huffman's algorithm is that it produces an *expression of minimum value*, from among all expressions on the given elements $\{a_1, \ldots, a_n\}$, whenever axioms A0–A4 hold. We can prove this by starting with any expression $E$ and transforming it into expressions of equal or lesser value until we obtain the result of Huffman's construction. First we let $a_i$ be the smallest of $\{a_1, \ldots, a_n\}$. If $l_i < \max(l_1, \ldots, l_n)$ (i.e., if $a_i$ is not at the deepest level of $E$), let $a_k$ be an element at the deepest level; then some ancestor of $a_k$ is at the same level as $a_i$, and we can transform $E$ into $E' = E$ where $a_i$ is a brother of this ancestor. One of the nephews of $a_i$ in $E'$ is $a_k$ or an ancestor of $a_k$; call it $x$. The other nephew, call it $y$, is not. Since $y$ has been computed from one or more elements greater than or equal to $a_i$, we have $a_i \le y$ by axiom A0. Thus $(a_i \circ x) \circ y \le a_i \circ (x \circ y)$; replacing $(a_i \circ (x \circ y))$ by $((a_i \circ x) \circ y)$ in $E'$ yields an expression $E'' \le E'$, because of axiom A3. Furthermore $a_i$ has moved to a deeper level in $E''$, while $a_k$ is still at the same level, which is still maximum among all levels. After repeating this transformation enough times, $a_i$ will appear at the deepest level. The same process can now be repeated with respect to the second-smallest element, $a_j$, this time using $a_i$ instead of $a_k$ in the argument. Finally, with both $a_i$ and $a_j$ on the same level, we can make them brothers, and $E$ has been reduced to an expression $E'''$ containing $(a_i \circ a_j)$. Replacing $a_i$ and $a_j$ by $(a_i \circ a_j)$, we can repeat the process until the desired Huffman-expression has been reached.

It is not clear that axiom A0 is necessary for the validity of this result; however, Huffman's construction leads to trees of comparatively little interest if axiom A0 is violated, so there seems to be little harm in assuming A0. It can be shown that axioms A0–A4 do *not* imply the law

$$\text{if } x \le y \text{ then } \big((x \circ a) \circ b\big) \circ y \le \big((y \circ a) \circ b\big) \circ x,$$

although this seems but a mild extension of A4. Thus, if we are faced with an expression like $\big((a \circ b) \circ (c \circ d)\big) \circ e$ where $e$ is the smallest element, we cannot simply exchange $e$ with $d$, say, in an attempt to move $e$ to the deepest level; the argument in the previous paragraph used A0 to conclude that $c \circ d \ge e$, so that $e$ could be exchanged with $(c \circ d)$ via A4.

The fact that Huffman's algorithm produces the minimum expression on $\{a_1, \ldots, a_n\}$ does not obviously imply Huffman's original theorem that the minimum value of $\sum a_i l_i$ is obtained, when the $a_i$ are nonnegative real numbers and the operation $a \circ b$ is simply $a + b$. For whenever $a \circ b$ is associative, all expressions on $\{a_1, \ldots, a_n\}$ are equal. Previous papers about abstractions of Huffman's method have therefore worked with two separate operations, one for the values that control the construction of the expression and the other for the evaluation function that is to be minimized. However, it is possible to deduce Huffman's theorem without this extra apparatus, by defining a suitable nonassociative operator that works with pairs of numbers instead of single reals.

Let $A$ be the set of ordered pairs $(a, a')$ of nonnegative real numbers, ordered lexicographically so that $(a, a') \le (b, b')$ if $a < b$ or $a = b$ and $a' \le b'$. The operation

$$(a, a') \circ (b, b') = (a + b,\ a + b + a' + b') \tag{1}$$

is easily seen to satisfy A0–A4. Therefore, the result of Huffman's construction applied to given pairs $\{(a_1, a_1'), \ldots, \{(a_n, a_n')\}$ is an expression of minimum value. It is not hard to see that the value of any such expression with respect to this operator is the pair

$$(a_1 + \cdots + a_n,\ a_1 l_1 + \cdots + a_n l_n + a_1' + \cdots + a_n'),$$

where each $l_i$ is the level of $(a_i, a_i')$ as before. Since the first component is independent of the $l_i$, Huffman's construction does indeed minimize $\sum a_i l_i$.

Another interesting operation on pairs is

$$(a, a') \circ (b, b') = \big(\max(a, b) + 1\,,\ a'(a \geq b) + b'(b \geq a)\big)\,, \tag{2}$$

where '$(a \geq b)$' is 1 or 0 according as $a \geq b$ or $a < b$. This operation also satisfies A0–A4; for example, when verifying A2 we have

$$\begin{aligned}&\big((a, a') \circ (b, b')\big) \circ \big((c, c') \circ (d, d')\big) = \\ &\qquad \big(\max(a, b, c, d) + 2\,,\ a'(a \geq b, c, d) + b'(b \geq a, c, d) + c'(c \geq a, b, d) + d'(d > a, b, c)\big)\,,\end{aligned}$$

which is symmetrical in the four arguments. Huffman's construction produces the expression of minimal value, which in this case is the minimum value of

$$\Big(\max_{1 \leq j \leq n} (a_j + l_j),\ \sum\{a'_i \mid a_i + l_i = \max_{1 \leq j \leq n} (a_j + l_j)\}\Big).$$

It is interesting to search for additional operations that satisfy A0–A4, since each of these corresponds to a minimization algorithm. The quadruple operation

$$(a, a', a'', a''') \circ (b, b', b'', b''') = (a + b\,,\ a' + b'\,,\ a + b + a'' + b''\,,\ a' + b' + a''' + b''') \tag{3}$$

illustrates another possibility: the optimum in this case is

$$\big(\textstyle\sum a_i\,,\ \sum a'_i\,,\ \sum a_i l_i + \sum a''_i\,,\ \sum a'_i l_i + \sum a'''_i\big)\,,$$

so we minimize the 'weighted path length' $\sum a_i l_i$ and—among all trees for which *this* is minimum—we minimize another weighted path length $\sum a'_i l_i$.

At first glance, operations (1), (2), (3) may seem very tricky or mysterious or both. Actually there is a fairly simple way to account for all of them: the function

$$a \circ b = x(a + b) \tag{4}$$

satisfies A0–A4 for all $x \geq 1$, over the nonnegative reals. Operation (1) corresponds to the multiplier $x = 1 + \epsilon$, where the pairs $(a, a')$ correspond to polynomials $a + a'\epsilon$ in $\epsilon$, modulo $\epsilon^2$. Operation (3) is similar but with $x = 1 + \epsilon^2$. Operation (2) corresponds to large values of $x$; it records the degree and leading coefficient of a polynomial in $x$ so that $(a, a') \leftrightarrow x^a a' + O(x^{a-1})$. The author has been unable to construct an operation that corresponds to minimization of $\max(l_1, \ldots, l_n)$ over all expressions that minimize $\sum a_i l_i$; it seems that such an operation might exist, possibly a very simple one, because Schwartz [9] showed that a variant of Huffman's algorithm does this. The arguments of Parker [8], that isomorphic versions of (4) account for essentially all operations satisfying A0–A4, are incomplete, but it does appear that all of the useful operations are strongly related to (4).

When $a \circ b = a + b$, the formula $\sum_{1 \leq i \leq n} a_i l_i$ can be rewritten $\sum_{1 \leq j < n} s_j$, where $\{s_1, \ldots, s_{n-1}\}$ is the set of subexpressions of a given expression. For example, in the expression $\big(((a_1 + a_2) + a_3) + (a_4 + a_5)\big)$, we have $3a_1 + 3a_2 + 2a_3 + 2a_4 + 2a_5$ $(a_1 + a_2)$ + $\big((a_1 + a_2) + a_3\big)$ + $(a_4 + a_5)$ + $\big(((a_1 + a_2) + a_3) + (a_4 + a_5)\big)$. Hu and Tucker [4] proved that the sum of the first $k$ subexpressions formed by Huffman's algorithm is less than or equal to the sum of any $k$ subexpressions built up successively starting with $\{a_1, \ldots, a_n\}$ and replacing $a_i$ and $a_j$ by $a_i + a_j$. Glassey and Karp [1] showed that this has extensive consequences, for it implies that Huffman's algorithm minimizes not only $\sum_{1 < j < n} s_j$ but also $\sum_{1 < j < n} f(s_j)$ for any nondecreasing concave function $f$. These facts can be put into our algebraic framework in the following way.

Let $\bullet$ be an associative, commutative operator over $A$, satisfying the following operations:

**B1.** $(a \circ b) \bullet (c \circ d) = (a \circ c) \bullet (b \circ d)$.
**B2.** If $a \le b$ then $a \bullet c \le b \bullet c$.
**B3.** If $a \le c$ then $(a \circ b) \bullet c < a \bullet (b \circ c)$.

We can now show, by mimicking the previous proof in a straightforward way, that Huffman's procedure has the following strong property: Suppose that the first $k$ steps of Huffman's algorithm have reduced the initial elements $\{a_1, \ldots, a_n\}$ to the elements $\{a'_1, \ldots, a'_{n-k}\}$, and consider any other $k$-step process that obtains $\{a''_1, \ldots, a''_{n-k}\}$ by repeatedly choosing two elements $\{a_i, a_j\}$ and replacing them by $(a_i \circ a_j)$. Then

$$a'_1 \bullet \cdots \bullet a'_{n-k} \le a''_1 \bullet \cdots \bullet a''_{n-k}.$$

This generalizes our previous result, which was the special case $k = n - 1$. The lemma of Hu and Tucker follows by defining $a \circ b$ as in (1) and taking $(a, a') \bullet (b, b') = (a + b, a' + b')$. We can let $a \bullet b = a + b$ when $a \circ b$ has the form (4).

It is easy to see that the subexpressions produced by Huffman's algorithm are nondecreasing: If we number the $s_j$'s in the order they are created, we have $s_1 \le \cdots \le s_{n-1}$. Jan van Leeuwen [10] has exploited this to show that Huffman's procedure can be carried out in linear time, if we assume that the inputs are given in order $a_1 \le \cdots \le a_n$: After $k$ steps, the remaining $n - k$ elements will be $\{a_{i+1}, \ldots, a_n\}$ and $\{s_j, \ldots, s_k\}$, for some $i \le n$ and $j \le k$; initially $i = j = k = 0$. Then the smallest remaining element is $s_j$, if $i = n$, otherwise it is $\min(a_{i+1}, s_j)$; if it is $a_{i+1}$, we increase $i$ by 1, otherwise we increase $j$ by 1. The second-smallest element is then found in the same way, using $a_{i+1}$ if $j < k$. Finally $s_{k+1}$ is computed and $k$ is increased by 1.

Consider the behavior of this procedure in the case of operation (3), when $a_1 \le \cdots \le a_n$ and $1 \le a'_i \le 2$. Then $s'_j \ge 2$, so the comparison of $(a_{i+1}, a'_{i+1}, a''_{i+1}, a'''_{i+1})$ to $(s_j, s'_j, s''_j, s'''_j)$ can be based entirely on the first components $a_{i+1}$ and $s_j$, where we regard $a_{i+1}$ as smaller than $s_j$ in case of equality. The particular values of $a'_i$, $a''_i$, $a'''_i$ have no effect on the algorithm. It follows that van Leeuwen's efficient procedure can be used on the singleton elements $a_1 \le \cdots \le a_n$ instead of the quadruples of (3), with the tie-breaking rule that $a_{i+1}$ *should be preferred* to $s_j$ in case of equality; we obtain a binary tree (i.e., an expression) that minimizes $\sum a_i l_i$. Furthermore, among all binary trees that obtain the minimum of $\sum a_i l_i$, this one also minimizes $\sum a'_i l_i$ for all choices $1 \le a'_i \le 2$. (The special case $a'_i = 1$ for all $i$ was proved by Schwartz in [9].) This same binary tree also attains the lexicographic minimum of

$$\left(\sum a_i l_i, \sum a_i l_i^2, \sum a_i l_i^3, \ldots\right),$$

because van Leeuwen's algorithm will produce the identical tree when operation (4) is used with $x = 1 + \epsilon$, for all sufficiently small $\epsilon$, and we have $\sum a_i (1+\epsilon)^{l_i} = \sum a_i + \sum a_i l_i + \sum a_i \binom{l_i}{2} + \cdots$.

## References


[1] C. R. Glassey and R. M. Karp, "On the optimality of Huffman trees," *SIAM J. Appl. Math.* **31** (1976), 368–378.

[2] Martin C. Golumbic, "Combinatorial merging," *IEEE Trans. on Computers* **C-25** (1976), 1164–1167.

[3] T. C. Hu, Daniel Kleitman, and Jeanne K. Tamaki, "Binary trees optimum under various criteria," *SIAM J. Appl. Math.* **37** (1979), 246–256.

[4] T. C. Hu and A. C. Tucker, "Optimal computer search trees and variable length alphabetic codes," *SIAM J. Appl. Math.* **21** (1971), 514–532.

[5] David A. Huffman, "A method for the construction of minimum redundancy codes," *Proc. IRE* **40** (1951), 1098–1101.

[6] F. K. Hwang, "Generalized Huffman trees," *SIAM J. Applied. Math.* **37** (1979), 124–127.

[7] Alon Itai, "Optimal alphabetic trees," *SIAM J. Computing* **5** (1976), 9–18.

[8] D. Stott Parker, Jr., "Conditions for optimality of the Huffman algorithm," *SIAM J. Computing* **9** (1980), 470–489.

[9] Eugene S. Schwartz, "An optimum encoding with minimum longest code and total number of digits," *Information and Control* **7** (1964), 37–44.

[10] J. van Leeuwen, "On the construction of Huffman trees," *Proc. 3rd Int. Colloq. Automata, Languages, and Programming*, Edinburgh (July 1976), 382–410.